PySpark Resource-Availability Testing

Emerson Kiefer, Matthew Spahl,
Owen Tibby, Kevin Martell

WHO ARE WE?

Emerson Kiefer

Assisted with testing, validation and explaining the experimental results.
Created design diagrams and organized the presentation.

Matthew Spahl

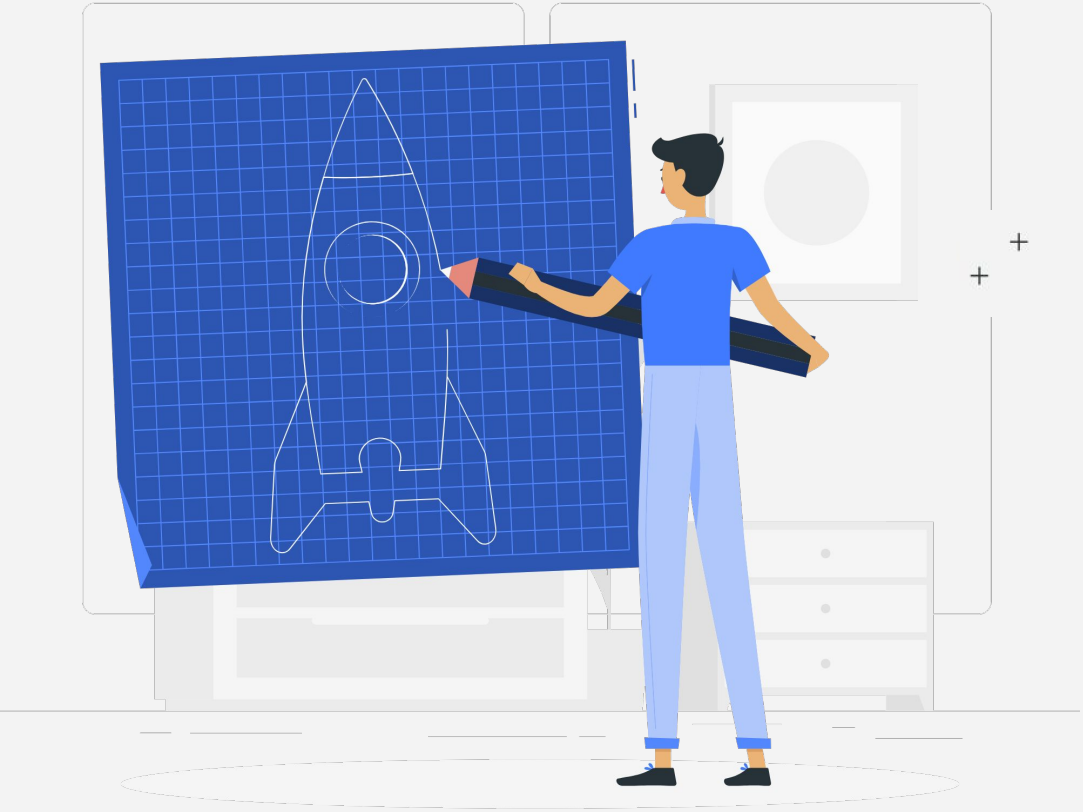
Coded PySpark random forest program, wrote instructions for running code, ran experiments/tests on VM, documented test results.

Owen Tibby

Sourced dataset, performed data cleansing/ preprocessing, assisted with logic for ML model, and designed visuals.

Kevin Martell

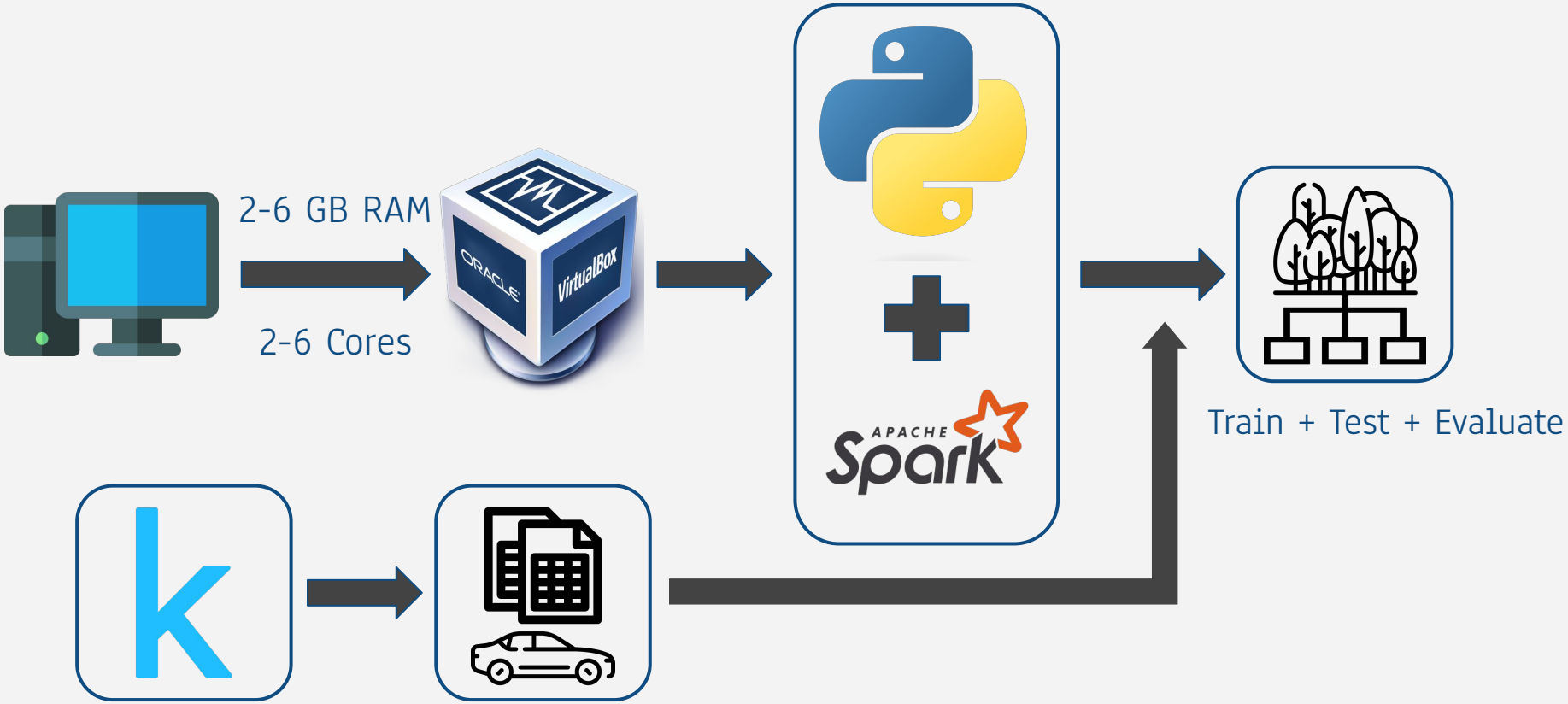
Assisted with cloud services research, narrowed the final project, ran pyspark random forest code in colab for testing purposes



(Icon made by Freepik from www.flaticon.com)

1: Design Description

Presented by Emerson



(Icons made by Freepik from www.flaticon.com)

PySpark_Forest.py

Preprocessing Time

Training + Prediction Time

cleaned_data_CS532.csv

Initial PySpark Dataframe

Index Categories

One-Hot Encoding

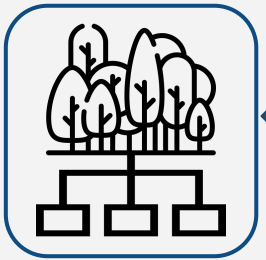
70% of data

Training Dataframe

30% of data

Testing Dataframe

Train Classifier



RandomForestRegressor

Hyper-parameter:

# of trees

Predict Test Inputs

Predictions Dataframe

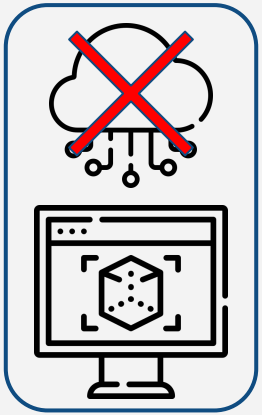
Evaluate Mod

RMSE

Testing Root Mean Squared Error

Total Time Elapsed

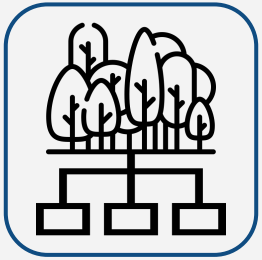
Design Decisions



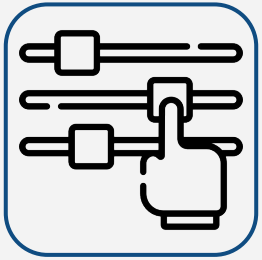
VM vs Cloud

PySpark

Used Car Dataset



Random Forest Regression



Hyperparameters

(Icons made by Freepik from www.flaticon.com)

2: Experiments/ Tests

Presented by Matt

(Icon made by Freepik from www.flaticon.com)

Experiment/Test Specifications

Trained and tested random forests by running program in virtual machine (VirtualBox and Ubuntu 20.04.5, validated on second Ubuntu VM)

Max tree depth of 7, and varied:

- **RAM** available to virtual machine (2 GB, 4 GB, 6 GB)
- **CPU cores** available to virtual machine (2 cores, 4 cores, 6 cores)
- **Number of trees** in the random forest (10, 50, 100, 150, 200)

Averaged over 3 trials each
Recorded total time elapsed: preprocessing time + training/predicting time

Experiment/Test Observations

- More than 200 trees with 6 cores -> program runs out of memory
- Pyspark random forest regressor likely uses parallelism (reference: https://spark.apache.org/docs/2.2.0/mllib-ensembles.html)
- More cores -> more data processed at once -> more memory / RAM needed
- Cache memory warnings (More warnings when more trees, disk reads slow training)

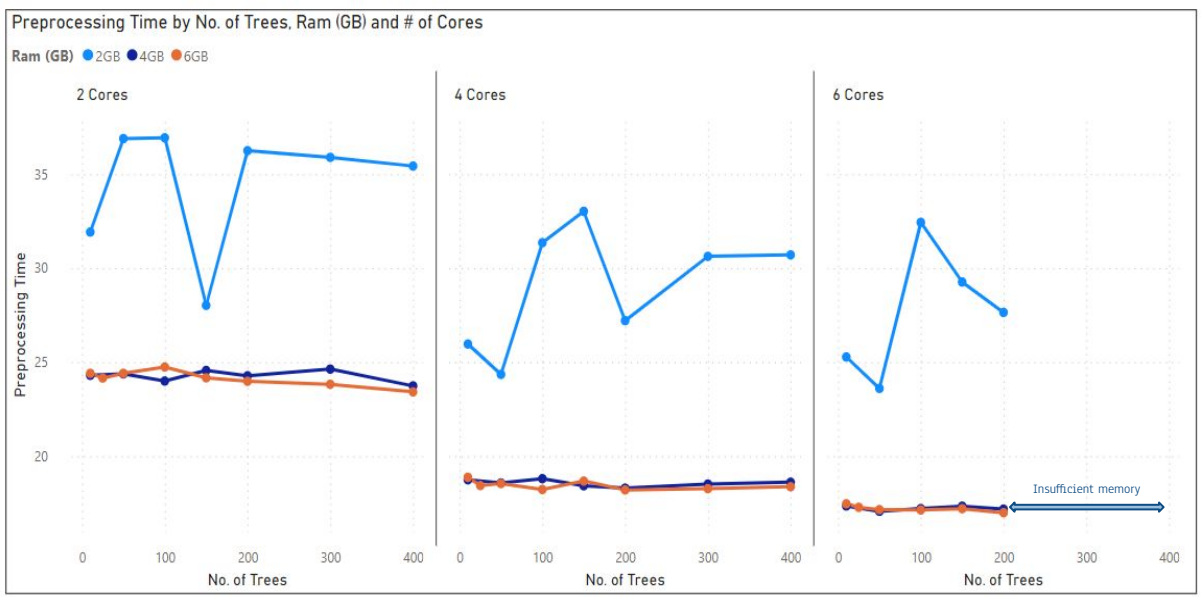
```
only showing top 20 rows

Size of training dataset: 311336
Size of test dataset: 133772
23/11/19 15:35:51 WARN MemoryStore: Not enough space to cache rdd_80_1 in memory! (computed 12.4 MiB so far)
23/11/19 15:35:51 WARN BlockManager: Persisting block rdd_80_1 to disk instead.
23/11/19 15:35:51 WARN MemoryStore: Not enough space to cache rdd_80_4 in memory! (computed 29.2 MiB so far)
23/11/19 15:35:51 WARN BlockManager: Persisting block rdd_80_4 to disk instead.
23/11/19 15:35:58 WARN DAGScheduler: Broadcasting large task binary with size 1362.4 KiB
+----------+------------------+
|RV_percent|        prediction|
+----------+------------------+
```

(Icon made by Freepik from www.flaticon.com)

3: Experimental Results

Presented by Owen

Fig. 3.1



- ↑ **RAM** from **2GB to 4GB** ↑ **preprocessing performance**
- ↑ **RAM to 6GB ≠ enhanced performance.**
- **More cores → faster runtime,** but **advantages ↓** as there are likely costs associated with **non-parallelizable preprocessing** tasks.

Fig. 3.2

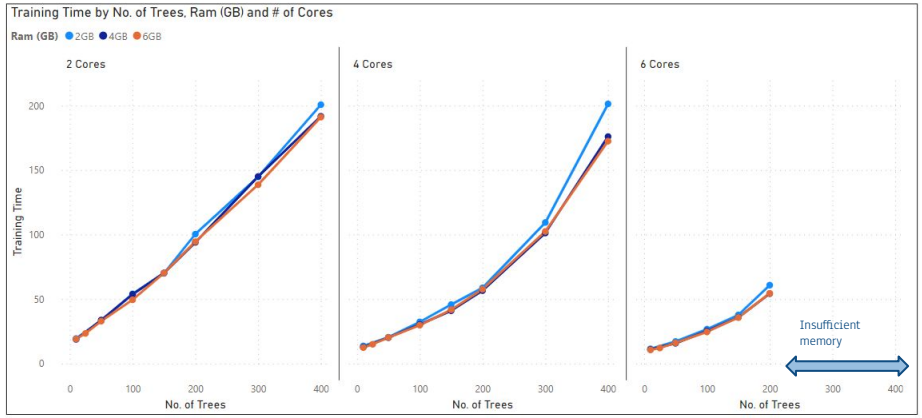


Fig. 3.3

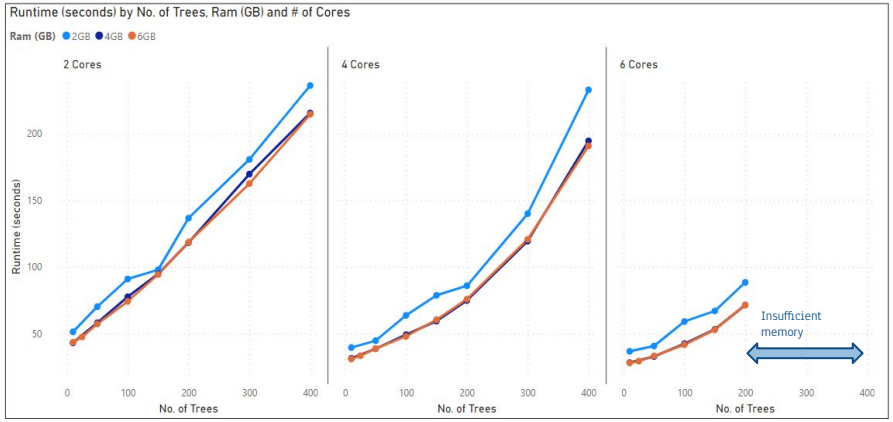
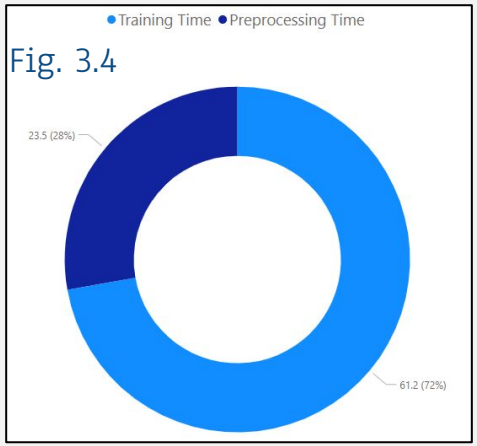
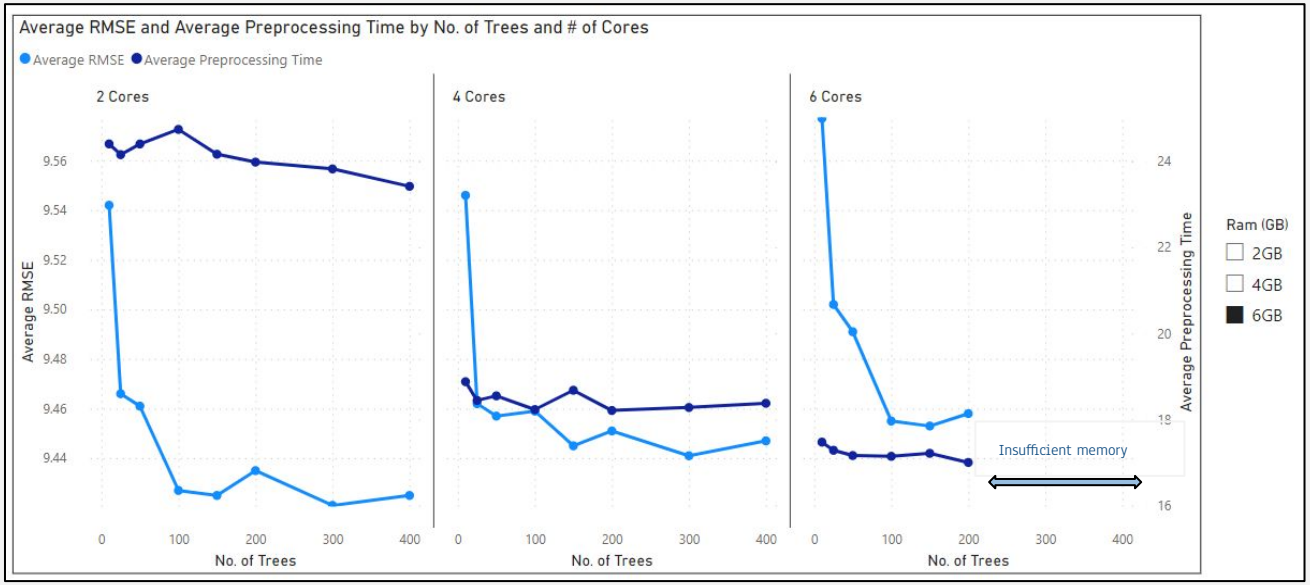


Fig. 3.4



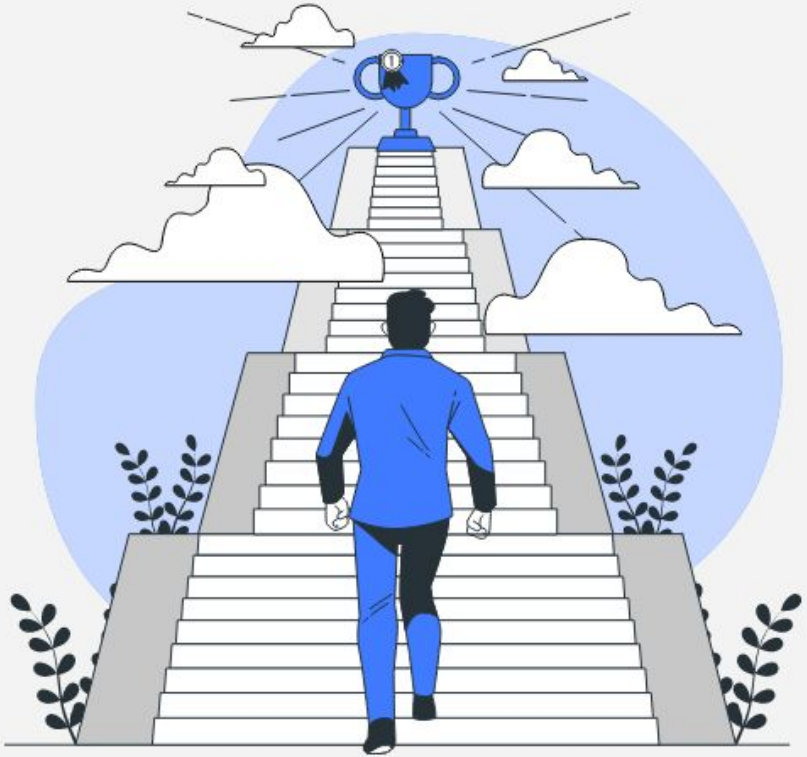
- 2 cores: Training $\propto$ # of trees (linear)
- 4 cores: Training $\propto$ # of trees^2 (quadratic), likely influenced by cache memory and RDD disk storage (slower reads)
- As training took up the larger share of runtime, the shape of the **total runtime graph** (Figure 3.3) closely **resembled** the **training time graph** (Figure 3.2).

Fig. 3.5



- With **RAM** fixed at **6GB, Figure 3.5 illustrates** general **RMSE convergence as more trees are added.**

- **Accuracy** **significantly** when using **> 100 trees.**



4: Goals

Presented by Kevin

Evaluated Hypothesis

Would there be a linear runtime increase with respect to the number of trees in the forest?

Runtime Experiments

Virtual Machine Configuration
Random forest Parameters
Testing scenarios

Visualize Performance

Created graphs to illustrate trends in performance based on the VM configuration and the model's hyperparameters

Explain Results

The parallelism introduced by using more cores increases memory requirements, impacting performance.

RAM Impact
CPU Cores Impact
Forest size Impact

(Icon made by Freepik from www.flatcon.com)

5: Possible Improvements

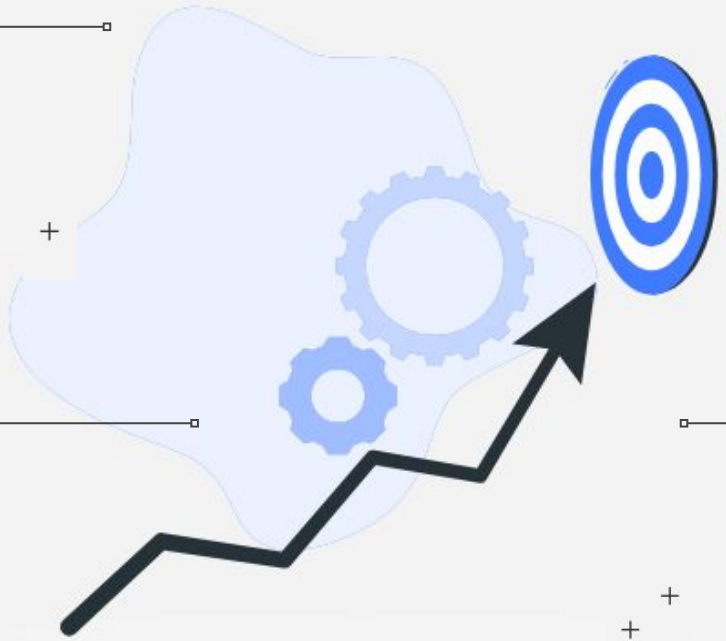
Presented by Kevin

GPU Tests

Compare tradeoffs between GPU parallelization benefits and CPU optimized libraries

Cloud Tests

Compare performance in a Cloud environment. Consider the impact of network latency and server performance



Memory Warning Analysis

Use PySpark's logging system to evaluate the correlation between cache memory warnings and runtime

Tree Depth Experiments

Run all experiments again with different settings for tree depth to see how scaling of runtime changes

THANK YOU!

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, infographics & images by **Freepik,** and illustrations by **Storyset**

Please keep this slide for attribution